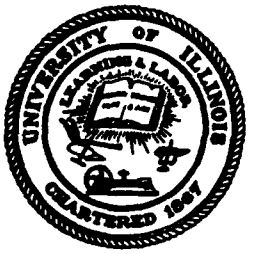
(2)

ADVANCED CONSTRUCTION TECHNOLOGY CENTER
Document No. 89-37-04



AN OBJECT-ORIENTED ENVIRONMENT FOR REPRESENTING BUILDING DESIGN AND CONSTRUCTION DATA

By

J. H. Garrett, Jr.
J. C. Basten
J. F. Breslin

Department of Civil Engineering
University of Illinois at Urbana-Champaign

A Report of Research
sponsored by the
Army Research Office

ADVANCED CONSTRUCTION TECHNOLOGY CENTER
NEWMARK CIVIL ENGINEERING LABORATORY
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
205 NORTH MATHEWS AVENUE
URBANA, ILLINOIS 61801

JUNE 1989

Advanced Construction Technology Center
University of Illinois at Urbana-Champaign

The Advanced Construction Technology Center is a unit of the College of Engineering. It is administered by the Department of Civil Engineering. The Army Research Office provides core financial support under the Department of Defense--University Research Initiative Program. The Center's multidisciplinary research program is conducted by the faculty and graduate students of the University of Illinois at Urbana-Champaign.

United States Army representatives to the Center include Dr. Fritz Oertel, Scientific Program Officer for the Army Research Office and the following other members of the U.S. Army Advisory Committee to the Center.

Dr. L.R. Shaffer, U.S. Army Construction Engineering Research Laboratory, Champaign, Illinois (Committee Chairman);

Dr. Eugene Marvin, U.S. Army Cold Regions Research and Engineering Laboratory, Hanover, New Hampshire;

Dr. Margaret E. Roylance, Composite Materials Technology Laboratory, Watertown, Massachusetts;

Dr. Robert Storer, Naval Civil Engineering Laboratory, Port Hueneme, California;

Dr. Lillian D. Wakeley, Waterways Experiment Station, Vicksburg, Mississippi;

Dr. Guiliano D'Andrea, BNT Laboratory, Watervliet Arsenal, Watervliet, New York.

The Center Director is Professor Joseph P. Murtha, Newmark Civil Engineering Laboratory, University of Illinois, 205 North Mathews Street, Urbana, Illinois 61801, phone number (217)333-6937.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

MASTER COPY - FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARO 24605.39-EG-UIR
6a. NAME OF PERFORMING ORGANIZATION U of Illinois at Urbana-Champaign 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U. S. Army Research Office
6c. ADDRESS (City, State, and ZIP Code) 205 North Mathews Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAL03-87-K-0006
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
An Object-Oriented Environment for Representing Building Design and Construction Data

12. PERSONAL AUTHOR(S)
J. H. Garrett, Jr., J. C. Basten, J. F. Breslin

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	June 1989	35

16. SUPPLEMENTARY NOTATION
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Building Design, Construction Data, Architecture, Building Models

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report describes an architecture for a building modelling environment that represents the functional and spatial decompositions of a constructed facility. The architecture, based on object-oriented methods, consists of collections of functional and spatial objects describing the various levels of functional and spatial decomposition of a building; the interrelationships between these objects are represented via several types of constraints — functional, spatial and form-function constraints. The benefits of this model, as opposed to more traditional

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

CAD building models, is that information other than geometric and material properties is represented; this model is capable of identifying the physical objects (geometry, location and material), how they fit into functional systems within the building, how their functional attributes are influenced by form and other functional systems, and of which abstract spaces (rooms, floors, etc) they are a part. Although small portions of this architecture have been implemented to test the concepts described in this report, implementation of this modelling environment has yet to be completed.

An Object-Oriented Environment for Representing Building Design and Construction Data

Final Report
Project # 1-5-25950-37

J. H. Garrett, Jr.
Principal Investigator

J. C. Basten
Research Assistant

J. F. Breslin
ACTC Fellow

Advanced Construction Technology Center
University of Illinois at Urbana-Champaign
Urbana, IL 61801

March 15, 1989

Interim Technical Report

This is a interim technical report for the Advanced Construction Technology Center project titled *Unified Knowledge-Based Model for Building Design and Construction*, project number 37. Professors L.A. Lopez and J.H. Garrett, Jr. were the project investigators. This reproduction is provided for other researchers, particularly those researchers in Army Laboratories interested in the work.

Forward

This report describes the work performed on ACTC project # 1-5-25950-37 entitled "An Object-Oriented Model for Building Design and Construction", which was funded by the Advanced Construction Technology Center at the University of Illinois at Urbana-Champaign.

This project was a small initial effort addressing the much larger, more ambitious goal of developing a flexible data model that will serve to integrate the design and construction industry participants. This project did not intend to develop a CAD environment; the intent was to develop a concept for a data model, based on object-oriented programming, that would serve to further guide system and model development.

Abstract

This report describes an architecture for a building modelling environment that represents the functional and spatial decompositions of a constructed facility. The architecture, based on object-oriented methods, consists of collections of functional and spatial objects describing the various levels of functional and spatial decomposition of a building; the interrelationships between these objects are represented via several types of constraints — functional, spatial and form-function constraints. The benefits of this model, as opposed to more traditional CAD building models, is that information other than geometric and material properties is represented; this model is capable of identifying the physical objects (geometry, location and material), how they fit into functional systems within the building, how their functional attributes are influenced by form and other functional systems, and of which abstract spaces (rooms, floors, etc) they are a part. Although small portions of this architecture have been implemented to test the concepts described in this report, implementation of this modelling environment has yet to be completed.

1. INTRODUCTION

The fragmentation of our national construction industry has been well documented and given much of the blame for the decrease in its productivity and ability to compete in foreign markets [6]. Many believe that the solution to decreasing productivity of the construction industry is automation — of design, management, and construction activities. While the automation of specific tasks within the process will improve productivity, it is not a complete solution. Although the automation of many tasks within the building design and construction process is in progress (systems now exist that assist with the preliminary design of the structure[10], the detailed design of a structure[4], and the verification of design standards[2], to name a few), these systems represent "islands of automation" and still do not contribute to an *integrated* solution to the design and construction process. At this point, several questions to ask are: "What does integration mean?" and "Why is integration a necessary goal for improving productivity in the construction industry?"

1.1. What Is Integration?

Instead of giving a definition of integration, a performance-based definition of an integrated environment is presented. An integrated decision-making environment would provide to each of the participants of the building design and construction process:

1. access to all of the decisions (i.e., results of design) about the design/construction process made to date, from which a participant can acquire information required for him to do his task;

2. the ability to retrieve, or ask in a fairly responsive fashion, why certain decisions made by upstream participants were made and if there is any room for negotiation (maybe only the latter part is really important if one has access to the actual designer); and

3. the ability to ask, again in a fairly responsive fashion, if the decisions made by a participant appear to be reasonable to downstream participants (for example, getting a contractors opinion about constructibility problems);

Note that the above described definition of integrated decision making does not mention the computer. The key idea, in all three steps is communication — communication of decisions, communication of which decisions are negotiable, and communication of foreseen problems between decisions and downstream considerations.

1.2. Why Is Integration a Necessary Goal?

The benefit of this improved communication between participants is not the elimination of iteration (which step 2 helps make easier), but the elimination of **needless** backtracking that could have been avoided with better communication mechanisms. All participants will be able to make much more informed decisions, where the information on which they are basing

decisions comes from upstream participants (previous design decisions), midstream participants (decisions being made concurrently that may conflict), and downstream participants (constraints or identified violations presented by downstream participants reviewing the current design activity). Hence, an integrated environment essentially connotes an environment supporting communication of large amounts of information. Communication supports more informed decision making, which reduces iteration (but does not necessarily eliminate it). Reduced iteration, especially between the design and construction stages, leads to a higher productivity.

One of the major problems with current CAD systems is that they represent what the building looks like (quite well, in fact), but do not capture any other information about the building, such as the functional aspects of its subsystems nor the interrelations among these various systems. In fact, they sometimes serve to confound integration by restricting the information communicated among the participants. Thus, what is needed is a richer model of building information that serves to improve the communication among the participants by representing more than just geometric information in a form that is usable by all participants. The above described model does not exclude the use of non-computer-based models, but because of the amount of information that must be communicated, a computer environment is felt to be the only possible solution. The inverse of this point (and part of the reason CAD has not improved the productivity of the construction industry as was promised) is that so far we have been using the computer and its information processing capabilities as a "silicon substitute" for the paper-based drawings that communicated information for thousands of years before the computer. Hence, at this point a vital question to ask, which happens to be the essence of this research project, is: "What information needs to be represented (in the computer), and hence transferred from one participant to another?"

1.3. What Information needs to be Communicated?

It is true that every piece of building information need not be made visible to all other participants; on the other hand, certain groups of participants could make good use of each others detailed decisions (e.g., HVAC, plumbing and electrical designers). Hence, the architecture of a building model on which the integration of the participants can be based (shown in Fig. 1.) must consist of multiple levels of building information, where the inner-most model contains information fundamental to all participants (such as geometric, material, and functionality information) and the outer-most level contains information that is important to more than one participant. Note that this architecture is an extension of the KADBASE global model concept, described in [5], where the computer systems supporting various participants communicate with the appropriate databases through a single global model.

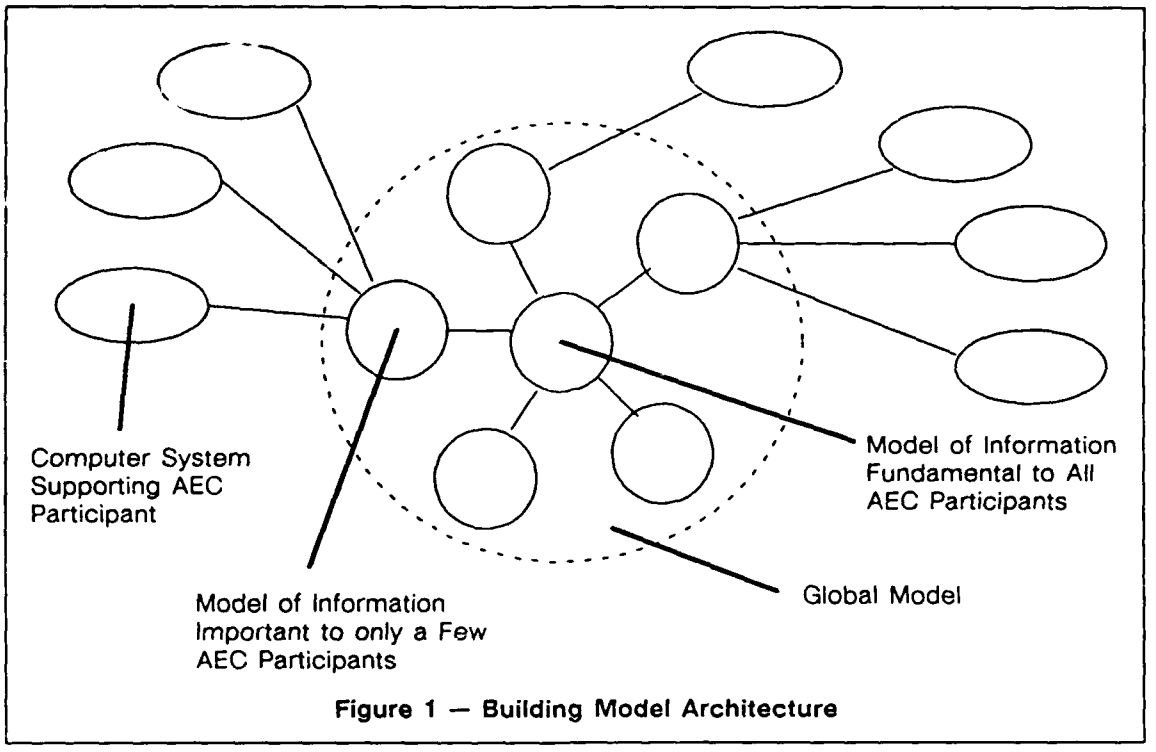


Figure 1 — Building Model Architecture

This report describes a project whose objective was to determine the content of, and develop an object-oriented model of, the core of the building information model shown in Fig. 1. Hence, this project focused only on representing the information that was felt to be important to all participants in the building design and construction processes. While other information is important to several participants, such as the activities derived and used to schedule the construction process, representation of this type of information was not addressed in this project.

2. BACKGROUND

2.1. Applicable Previous Work

In [1], Eastman defines a general organization for an integrated design database. According to Eastman, "The most convenient general conceptual organization of a project database is as a description of entities and their composition.[1]" He goes on to state that the representation of entities and their attributes alone is not sufficient for design; in addition to the entities, both functional and spatial compositions must also be represented because the individual parts may not represent the attributes of the composition, referred to a "emergent properties" by Eastman [1]. Hence, a more complete description consists of the entities, the functional composition and the spatial composition. The conceptual model that Eastman developed for structuring design information is termed an *abstraction hierarchy*. The research described in

this paper is an attempt to extend the abstraction hierarchy approach described in [1], and apply it as the organizational structure for the conceptual global model of a building.

Law also describes the application of the abstraction hierarchy approach to the representation of building design information in [9]. In that work, he describes a three part model made up of the following three parts: a topological hierarchy, a structural hierarchy and an architectural hierarchy. While being a correct and complete set of hierarchies for architectural and structural design, they do not form a complete model that would support all agents in the building design and construction process.

Keirouz developed an object-oriented (described in the next section) building model for use in robotic construction planning, that is in many ways similar to the model developed in this research. The model consists of functionally and spatially interrelated physical objects [8]. Each object maintains information about its geometry, non-geometric attributes, and its function within the same object. However, functional systems were not explicitly represented as objects and only existed implicitly as a set of interrelated objects. Keirouz places both functional and spatial relationships between two physical objects in what he calls a connection object.

Fenves, et al., developed an Integrated Building Design Environment (IBDE) that integrates several previously developed standalone knowledge-based systems that addressed the stages of building design: spatial layout, preliminary structural design, detailed standard-driven design, and construction planning [3]. The processes communicated in two ways: through status messages and a global building model. The global building model is a frame-based representation of the building, where the frames may represent very high-level abstractions (e.g., building) or very detailed descriptions of components (beam1). The global model developed for the IBDE is intended to capture the relevant information known about the building design that is used by downstream participants or critics. Note, however, that no mention of how geometry is dealt with in this model was mentioned, it can only be assumed that it was treated parametrically, i.e. as slots in frames. At certain levels of abstraction, this form of geometric representation is adequate. However, for a more detailed description of geometry, a more powerful representation is needed — such as boundary representation.

KADBASE (Knowledge Aided Database Management) is a prototype system for interfacing knowledge-based expert systems (KBESs) with database management systems (DBMSs) developed by Howard and Rehak[5]. The objective of KADBASE is to provide a flexible, knowledge-based interface in which the multiple databases can communicate as independent, self-descriptive components within an integrated engineering CAD system environment operating in a distributed computing environment.

In KADBASE, mappings between KBESs and DBMSs are divided into semantic mappings between data representations and syntactic mappings between data manipulation languages.

- **Semantic mappings** — mapping between data representations is a semantic transformation. The individual KBES may use different organizations for equivalent data or represent the data at different levels of abstraction. KADBASE integrates the dataspace of the KBESs through a global model based on frames or schemata. A semantic mapping consists of three parts: the local data representation, the expression of local data in the global model, and the global model.

- **Syntactic mappings** — mapping between database manipulation languages is basically a syntactic translation. This is essentially the concept of translating between one query language and another. KBES data requests are translated into a global syntax from the local syntax and then translated into the local syntax of the database management systems possessing the required information.

The KADBASE architecture consists of three principal components as shown in Figure 2. and described below.

- **Knowledge-Based System Interface** (KBSI) — a part of each KBES. It formulates the queries sent to, and processes replies sent from, the network data access manager (described next). It possesses knowledge of the local data model and performs the syntactic translations for queries and replies.

- **Knowledge-Based Database Interface** (KBDI) — an intelligent front end for a standard DBMS. It accepts queries from the network data access manager and returns the appropriate replies. Like the KBSI, it possesses knowledge of the local DBMS schema and handles syntactic translations of queries into local syntax.

- **Network Data Access Manager** (NDAM) — the actual interface between the KBESs and the DBMSs. It receives, translates into the global data model, and decomposes queries from the KBSIs, issues queries to the KBDIs, and formulates the replies to the KBSIs in the context local data model.

2.2. Object-Oriented Modelling Techniques

Object-oriented programming (OOP) is a programming methodology, and not a specific language. The basic building block of an object-oriented program is an *object*, a collection of data and methods for manipulating that data. The only method of communication between one object and another is *message passing;* one object sends a message to another object to provide a piece of data or execute a method (procedure or function).

In object-oriented environments, everything is an object. Objects can represent concepts, physical objects, processes, etc. In all cases, the object possesses a set of attributes and methods. Attributes represent data about the object; methods represent processes that the object is capable of performing. Attributes and methods are usually both represented as *slots* within the object. Other objects can access these slots, but only by sending a message to the

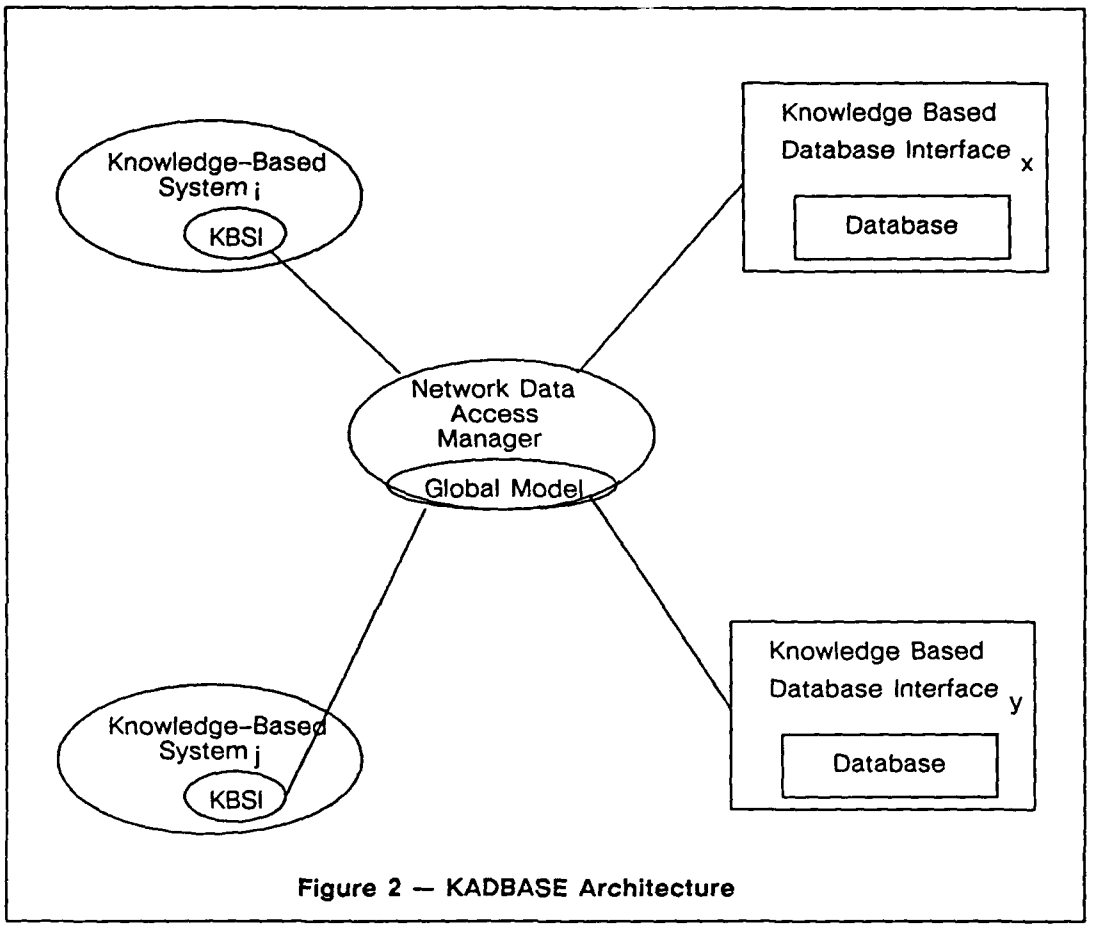


Figure 2 — KADBASE Architecture

object that "owns" the data or method. In addition, the attributes of an object may also have descriptive information, such as range or type. This information is stored in *facets* that are associated with the slots. Figure 3 shows the typical structure and an example of an object. In that example, the first three slots store declarative information about the constraint object, such as its expression or status. The "Expression" and "Status" slots have facets that restrict the values that the slots can have. The last slot stores the evaluatable form of that object, plus has a facet that can be messaged to generate that form from the expression stored in the "Expression" slot.

A common practice in object-oriented programming is to develop templates for types of objects, commonly called *class* objects. These class objects usually possess attributes, attribute values, method names, or methods that are common to several more specific objects. If these more specific objects are themselves templates for other even more specific objects, they are called *subclass* objects. Objects that represent an instance of a class or subclass object are called *instances*. For example, beam#2 could be an instance of the object wide-flange-beam, which is a subclass of the object steel-beam, which is a

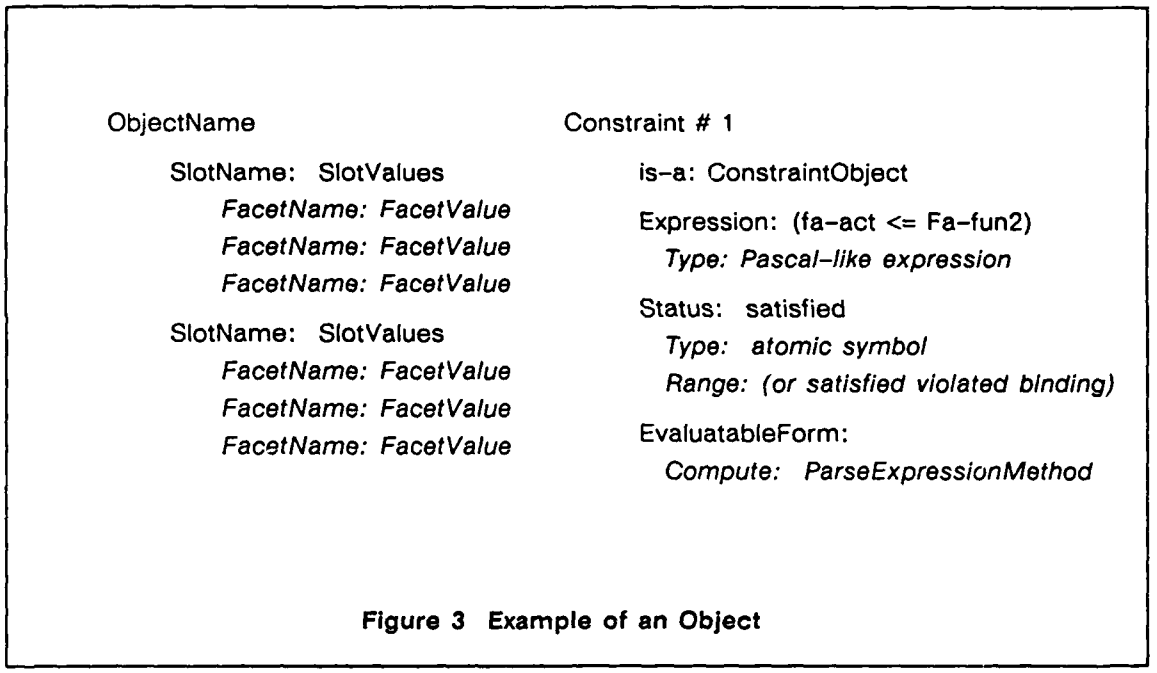


Figure 3 Example of an Object

subclass of the object beam. The object beam is called the *parent* of steel-beam, and steel-beam is the *child* of beam. These parent-child relations are important because in most object-oriented programming environments, children automatically *inherit* attributes and methods from parents. For example, all instances of the object ConstraintObject, like Constraint #1, will inherit the slotnames "Expression", "Status" and "EvaluatableForm", and their facets, from the object ConstraintObject. Through inheritance, it is possible to represent information at an appropriate level of object generality and have all more specific instances of objects inherit that information, thus reducing redundancy of information.

Another feature of object-oriented programming environments is the concept of a demon or active value, which basically watches a slot value and executes a method when that value is added, changed, or erased. These demons are also referred to as procedural attachments. This feature of object-oriented environments is especially suited for the expression and propagation of constraints.

Hence, the key ideas of object-oriented programming are that objects possess attributes and methods, can inherit attributes and methods from other objects, and communicate with each other (i.e., get data or execute an object's method) only by sending messages.

3. INFORMATION CONTENT OF CORE BUILDING MODEL

As was described in the introduction to this report, one objective of this project was to determine the fundamental information requirements of the various participants of the design

and construction processes. By fundamental information requirements, we refer to information that is usable by a large number of the participants in the building design and construction process. For example, while both construction managers and contractors will be concerned with the activities with which the building will be constructed, few other participants in the process are directly concerned with this information and thus it should be relegated to an outer layer in the model rather than within the core. The kinds of information deemed fundamental and belonging within the core model are described below.

1. The form, location, and orientation of all of the physical objects comprising a building. This requirement stems from the fact that a building is a physical object, made from smaller physical objects, and all physical objects have finite volume and uniquely exist at a spatial location and orientation.

2. Related to the first category, all physical objects are composed of some material. Hence, the material composition and properties of the physical objects are considered fundamental pieces of information. Cost estimators, structural engineers, and construction process planners all use material information.

3. A hierarchical decomposition of the space within a building and an identification of the physical components within, or bounding, that space would be used by almost all design and construction participants. In fact, spaces such as floor space, rooms, hallways, etc., may be conceived of before the physical objects that define them. These types of spaces are often used as grouping mechanisms and referred to by all participants.

4. A hierarchical decomposition of the functional systems within a building would be generated by designers and referred to by construction and facility managers; the functional systems can be further subdivided into functional subsystems, which themselves can be further subdivided. At some point, a primitive function (such as "beam") can be assigned to a physical object (object having spatial and material properties) within the building. In this way, the physical objects comprising the building can be categorized according to the functions they perform — i.e., of which functional systems and subsystems are they a part. This treatment facilitates the representation of multiple functional views over the same set of physical objects.

5. The relationships between the subsystems of one system (say HVAC) and the subsystems of another system (say power distribution) would be extremely useful for concurrent designers to represent their expectations of the other and would be useful for construction managers in determining dependencies which affect construction activity identification. In other words, how are physical objects functionally and spatially related? For example, if an HVAC system increases in size, how does it affect the structural system?

The functional decomposition of a building and the interrelation of the subsystems would be very useful for designers; it would permit them to know how the designs of other designers were intended to function. It is vital that construction planners and constructors know what the designer functionally intended with the collection of physical objects he has produced as his design — i.e., what functions do *they* perform and what and how are *they* related? Just as an

example, if given only the geometry and material of a wall in a building, a facility manager is unable to determine if it is a load bearing wall or not. If, on the other hand, the wall is represented as being part of a higher level structural system intended to resist gravity forces, such a mistake is much less likely to happen. *This functional information is not obvious from just looking at the geometric and material descriptions.*

Hence, the information content of the core building model should consist of the form, location orientation, and composition of the physical objects in the building, an identification of their function and how that function combines into larger functional systems, and an identification of the spatial decomposition of the building that is eventually defined by the physical objects.

4. OBJECT-ORIENTED BUILDING MODEL

Once it was determined what information should be represented within the core of the building modelling environment, the second objective of this project was to determine how that information could be represented using object-oriented methodologies.

4.1. Core Model Architecture

In order to meet the representational objectives of the core model as described in the previous section, an object-oriented model is proposed that consists of seven major types of objects:

1) objects for describing the functional primitives, functional subsystems and functional systems that are present within an building;

2) objects for describing the functional interrelationships between the attributes of one functional system and the attributes of another — functional constraints;

3) objects for describing the spatial primitives (space that is undivided and composed of a single material) and composite volumes (aggregations of spatial primitives), including objects for describing the features (surfaces, edges and vertices) of these spaces;

4) objects for representing more abstract spaces within the building, such as floors, rooms, halls, wings, sections, etc., called spatial systems and subsystems;

5) objects for describing the spatial interrelationships between the features of one spatial primitive and the features of another spatial primitive — spatial constraints;

6) objects for describing the interrelationships between functional system attributes and spatial system attributes — form-function constraints; and

7) objects for describing various materials out of which building components are made.

As described previously, an object is a collection of slots that represent either attributes or methods that can be performed by the object. A model of a specific building is constructed by

defining instances of predefined functional system objects, creating instances of spatial objects, and defining a variety of constraint objects that relate the functional and spatial attributes of the objects composing the building.

4.2. Functional System Objects

Buildings are composed of many functional systems that support all of the intended functions of the building, such as protection, comfort, lighting, performance of specific tasks, etc. These functional systems are composed of subsystems that are eventually composed of primitive functional objects, such as a beam. Each of these functional system hierarchies represents a different functional perspective of the building. For example, the structural engineer looks at the superstructure, which may be comprised of frames that are at the lowest level made up of steel sections. This is his perspective of the building, but not that of the HVAC engineer or anyone else. The HVAC engineer views the building as a collections of ducts, compressors, HVAC loads, etc. Hence, a functional description of a building can be described hierarchically where the higher levels describe the major functional systems that exist within the building and the lowest levels in the hierarchy describe the functional primitives.

4.2.1. Functional Primitives

Functional primitives are objects for which no further functional decomposition is practical, such as a beam, column, or pipe. Functional primitives are combined to form higher-level functional subsystems. The functional primitives composing a functional subsystem, besides being related to the subsystem, are interrelated to each other. This interrelationship is captured in the model proposed through the use of functional constraints (described in a later section).

These functional primitives possess attributes (represented as slots in the functional primitive objects), which are used to characterize their function. For example, a beam would have functional attributes such as magnitude and location of maximum deflection, the magnitude and location of maximum shear stress, and so on. Note, that any geometric attributes are not stored in the functional description of the functional primitive; these attributes are captured through the use of spatial primitives (described in a later section). Hence, only attributes relating to the function of a functional primitive are present within functional primitive objects.

To describe a functional primitive present within a structure, an instance of a functional primitive object (provided as part of the modelling environment) is created. The instance of the functional primitive object inherits all of the functional attributes defined for the functional primitive. These attributes, however, gain values that are specific to the building being modelled.

4.2.2. Functional Systems and Subsystems

As stated in the previous section, the functional systems are composed of functional subsystems, which are eventually composed of functional primitives. For example, the superstructure is composed of a lateral load resisting system and a gravity load resisting system, each of which are composed of other subsystems such as frames or shear walls. Hence, the functional systems and subsystems all have one attribute, *components*, which lists the components that make up the functional system. In this attribute are stored the subsystems and/or functional primitives that together make up the functional system.

A functional system may also have other functional attributes for which the value is dependent on the functional characteristics of its components. For example, the load carrying capacity of a frame is derived from the load carrying capacity of its members and their connections. This relationship between functional system attributes and the functional attributes of their components is again represented through constraints. As functional attributes of its components change, so must the dependent attribute of the functional system.

In addition to the relationships between functional systems and their components, it is also possible that one functional system may be interrelated to another functional system or systems. For example the load carrying capacity of a superstructure must be related to the load carrying capacity of the underlying foundation system. This type of relationship can also be represented using functional constraints.

4.2.3. The Complete Functional System Description

From the two types of functional system objects (functional systems and functional primitives), it is possible to represent the functional decomposition of a building. The "string" that ties these objects together into a complete functional description is the component attribute of functional systems and the use of functional constraints to define the interrelationships between the functional primitives, between the functional primitives and their aggregate functional system, and between functional systems. This hierarchical functional decomposition of a building is illustrated in Fig. 4.

As shown in Fig. 4, functional constraints may be written between the attributes of components and an attribute of a functional system thus representing a relationship between the system and its components. If any of the component attribute values change, the attribute value for the functional system changes. However, if the attribute of the functional system changes, the previously defined constraints are insufficient to determine how to modify the underlying component attributes. For example, the load carrying capacity of a frame is related to the load carrying capacities of its constituent beams and columns; if a beam or column

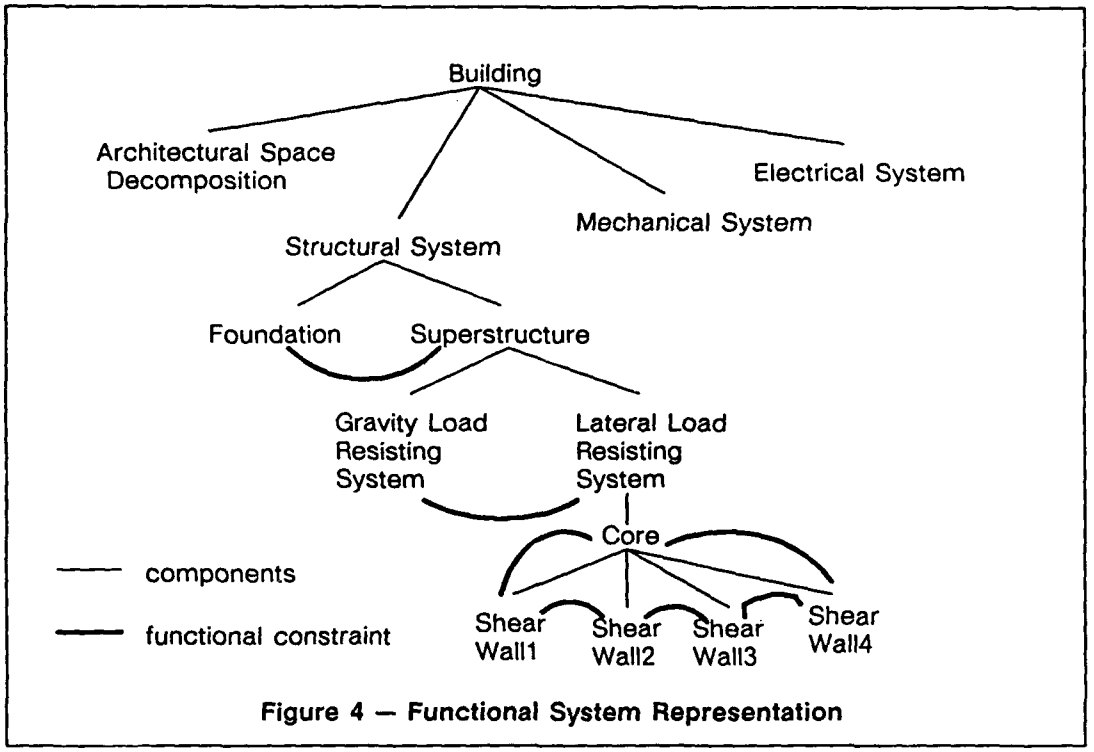


Figure 4 — Functional System Representation

load capacity is changed, the load carrying capacity of the frame is modified by the constraint linking component capacities to frame capacity. However, if the load carrying capacity of the frame is modified (say by some functional relationship to another functional system), how are the capacities of the components modified? There are a variety of ways in which the components can be modified to meet the new frame load carrying capacity specified. Hence, additional constraints stating which attributes of which components must be modified, and how, must be specified; these constraints thus link the attribute of the functional system to specific attributes of the components of the functional system. Note that the constraints being discussed thus far are unidirectional constraints, either in the direction from component to aggregation or from aggregation to component. Unidirectional constraints will be defined for the spatial system, as well.

4.2.4. The Usefulness of a Functional Decomposition

With a functional decomposition as described above, what could be inferred? One of the key uses of such a functional decomposition will be to provide a functional model through which the effects of changes in functionality can be inferred. By having the functional systems related to their components and to other functional systems, changes to the functionality of a component or a functional system can be propagated (via constraint propagation) to other functional systems and components. Such a model provides a powerful representational foundation on top of which could be built systems to assist in design and, more importantly, facility

renovation and modification. With the number of buildings that exist today that are in need of modification and renovation, having a model of the existing functional systems and their interrelationships would measurably assist in the tasks of modification and renovation.

4.3. Spatial System Objects

We have described how a building can be decomposed into its constituent functional systems and subsystems. Another way in which a building can be decomposed is spatially. The space within a building can, like the functional decomposition, be hierarchically decomposed into more and more detailed descriptions of space. At the lowest level of decomposition are spatial primitives — space that is not further subdivided and composed of a single material, such as steel or wood. From these spatial primitives, or their surface features, are composed the next level in the spatial decomposition, called composite volumes. All of the physical objects in a building are either spatial primitives (e.g., concrete block) or composite volumes composed of spatial primitives (reinforced concrete beam). The composite volume is used to represent an aggregation of primitives composed of various materials, e.g., a reinforced concrete column. Finally, spatial systems and subsystems represent the highest levels of abstraction for the description of spatial decomposition within a building (i.e., floors, wings, rooms, etc). Spatial system objects are used to *parametrically* describe an abstract space (i.e., length, width, height, or floor area) and to identify the spatial primitives and composite volumes that are within that abstract space.

4.3.1. Spatial Primitives and Features

The space within a building can, at the lowest level of decomposition, be thought of as "pieces" of building space that are unique in the three-dimensional space that they occupy and the material comprising them. These "pieces" are called spatial primitives. Spatial primitives thus describe the three-dimensional geometric properties of the space that they occupy and the material that occupies that space.

The geometric description of an object is achieved through several levels:

1) a complete three-dimensional solid model of the object;

2) an identification of the spatial features of the spatial object, such as surfaces, edges and vertices; and

3) a parametric description of *important* dimensions and other geometric attributes and the methods for computing these dimensions from the identified spatial features.

The representation of spatial information at levels 1 and 2 uses an architecture for geometric reasoning described by Woodbury [12], which consists of classes of spatial sets, features, abstractions, and constraints [1].

1) *Classes of spatial sets.* By class he means a template of attributes and methods that can inherit attributes and methods from other superclasses of templates. For example, the template for a cube would inherit attributes and methods from a template for a polyhedra. By spatial set he refers to the features, abstractions and constraints that are used to describe geometry.

2) *Features.* A feature is a reference to some part of a more complete geometric representation. The underlying geometric representation governs the types of spatial features to which a feature object can refer. For example, if the underlying geometric representation is a boundary representation (B-rep), then the features will be either vertices, edges, surfaces, or volumes. Fig. 5 shows an example of an underlying boundary representation geometric modelling object and the features to which reference might be made.

3) *Abstractions.* An abstraction can be thought of as a derived feature. If the underlying geometric representation does not directly support a desired feature, such as the centerline of a volume, then it must be computed from features that are available, such as the top and bottom surfaces.

4) *Constraints.* A constraint is an expression between attributes of features or abstractions that is maintained as changes are made to the geometry.

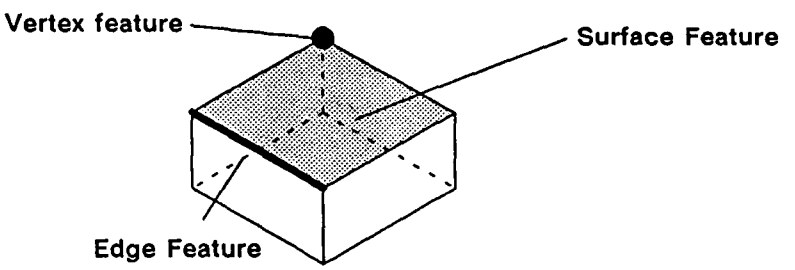


Fig. 5 — Sample Features of a B-Rep of a Cube

Such an architecture is designed to exist above a formal geometric modelling system, reference important features of the objects within the underlying geometric modelling object, and facilitate inferences concerning geometry. In short, geometry can be described and reasoned with using a collection of features, such as surfaces, edges, and vertices that are

1. In this work, the concept of abstractions is not utilized, but will eventually be incorporated. For now, the concept of abstraction is treated the same as a feature.

important for making spatial inferences about the object; the collection of *features* is referred to by Woodbury as a *class of spatial sets*. In this work, the classes of spatial sets are represented as spatial primitive objects that simply possess a reference to the underlying geometric modeling object and the important features of that underlying spatial description.

As an example, consider the cube shown in Fig. 5. The underlying boundary representation of the cube would represent each vertex, edge and surface in the cube and the topologic structure of these features. Hence, the geometric modelling object for a cube contains a vertex list, edge list, and a surface list. The ordering in the list has topologic meaning. A spatial primitive, for which the cube is a complete representation of its geometry, would possess an attribute that contains the name of this geometric modelling object. Important spatial features of this geometric modelling object to which the spatial primitive refers (for example, top-surface and bottom-surface) could be generated and made to reference the parts of the underlying geometric model object that represent the top- and bottom-surface; that is what features do — they refer to parts of the underlying geometric representation. These spatial features would then be placed in the "features" attribute of the spatial primitive, representing the spatial features of the spatial primitive that can be referred to. The presence of spatial features is very important for several reasons: 1) spatial constraints will be written between features of spatial primitives (discussed later) to describe how two primitives are spatially related, and 2) constraints will be written linking the *important* parameters, such as dimensions, of the spatial primitive to the attributes of these spatial features. Each of these spatial feature objects will possess methods for computing their own geometric dimensions and other information from the portion of the underlying geometric modelling object to which they refer. For example, a surface feature will be able to compute its surface area.

The third level of geometric description is accomplished through a set of parameters that describe the important dimensions of the spatial primitive. These parametric dimensions are used to generate the underlying solids model and are related to its resulting features. For example, the height and radius of a cone are used to generate the underlying solids model shown in Fig. 6. These two parameters are then be related to the resulting features of that cone; e.g., the height is equal to the distance, in the direction of the normal to the bottom surface of the cone, between the cone tip and the bottom surface (see Fig. 6). To ensure that the parameter values remain consistent with the features (which will be subject to movement due to the propagation of spatial constraints described later), parameter/feature constraints (described in a later section) must be defined to watch these two features for movement and recalculate the value of height accordingly. Because of the possibility of form-function constraints (described in a later section) being defined between the dimensional parameter (height) and some functional attribute, constraints must also be defined to watch the height

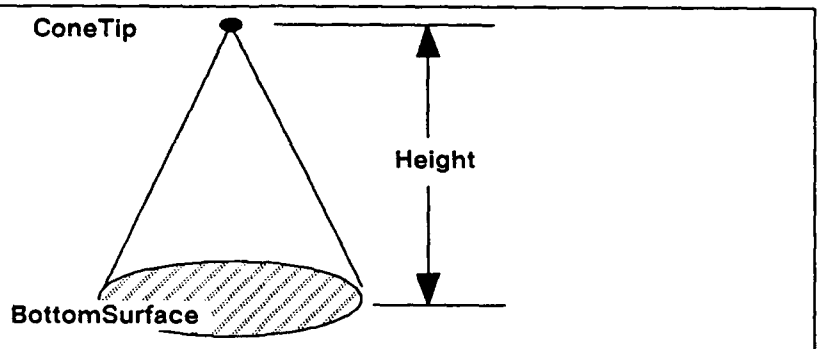


Fig. 6 — Relationship Between Parameters and Features

parameter for changes in value and move one or both of the features accordingly. Which features are moved depends on the defined relationship between the parameter and the features. For the cone, the change in height is shared by both the ConeTip and the BottomSurface. It may also be possible to ask the user which feature should be moved instead of presupposing a feature movement strategy.

Hence, each spatial primitive thus possesses a *reference* to the underlying complete geometric modelling object, a set of features that refer to parts of the underlying geometric model, and a set of dimensional parameters related to the features.

Note, that if the underlying geometric modelling environment is implemented in the same object-oriented environment as that in which the functional system and constraints are described, and the parts of the geometric modelling objects (vertices, edges and surfaces) are themselves represented as objects, the spatial primitive's feature objects *are* the vertex, edge, and surface objects of the underlying geometric model. What will happen in this case is that the objects used by the geometric modelling environment would be augmented with attributes and methods needed to describe the interrelationships between its features and dimensional parameters, and the objects used to describe the individual faces, edges and vertices would be given more meaningful names. If the underlying modelling environment is not founded in an object-oriented environment (like the VEGA environment used by Woodbury[12]), the feature objects are different from, and refer to, those objects in the underlying geometric modelling environment.

The material description of a spatial primitive, i.e., the description of a physical object, is achieved by identifying the material object describing the material composing the space. The spatial primitive then inherits all material properties from the identified material object. The materials from which building components are made is yet another hierarchy of objects

contained in this model. These objects, however, provide only static information about material properties and are very simply implemented in an object-oriented environment.

4.3.2. Composite Volumes

While the spatial primitives are used to describe the lowest level of spatial decomposition, the intermediate levels of spatial description are achieved through composite volumes. Composite volumes, as the name implies, are composed of other composite volumes or spatial primitives, meaning that all of their geometric properties are derived from the features of the component volumes. Hence, composite volumes have an attribute, components, in which the volumes that comprise the composite volumes are stored. Although composite volumes are aggregations of spatial primitives and other composite volumes, they differ from spatial systems (described in the next section) in that they still represent physical objects, albeit more complex physical objects.

The composite volume is generated by identifying the component volumes that comprise the composite volume and then describing the spatial constraints, such as attachment, between the features of these component volumes (see Fig. 7.). These constraints represent the

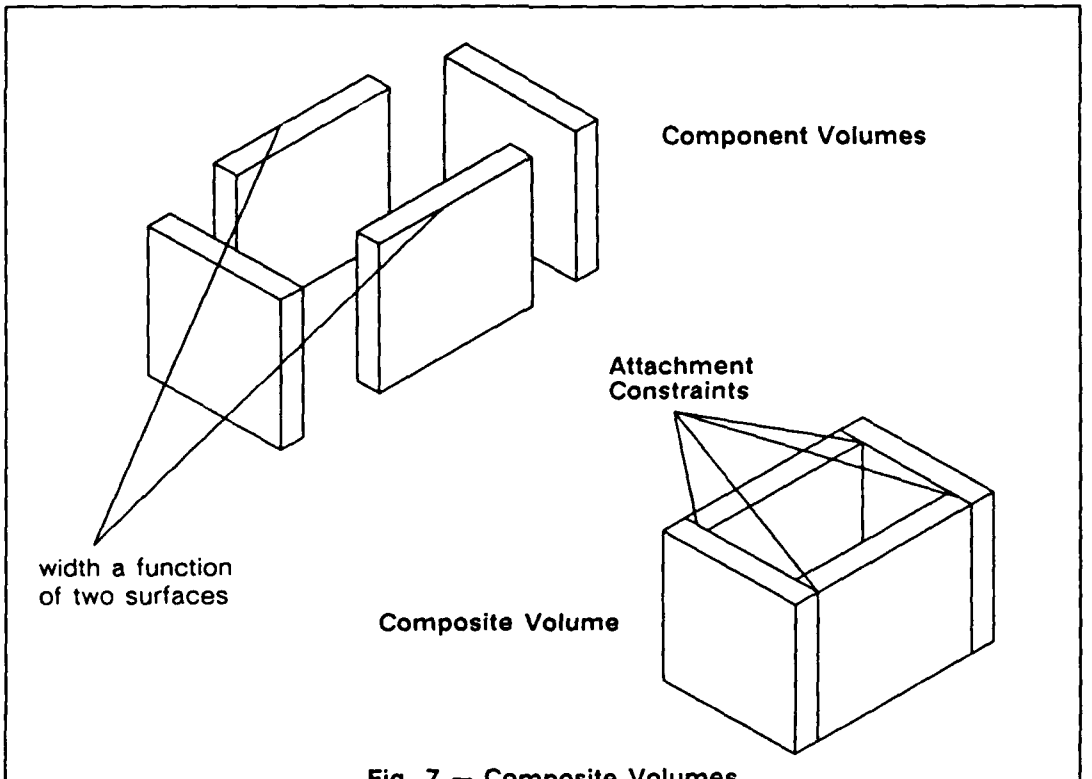


Fig. 7 — Composite Volumes

essence of the composite volume; simply identifying the components of a composite volume

does not state *how* the components interact to form the whole — this is the role of the spatial constraint. Any attribute associated with the composite volume is expressed in terms of the attributes of features of its component spatial primitives and the parameters of its component composite volumes. For example, the overall width of the composite volume shown in Fig. 7. is a function of the location of the outside surface features of the two opposing component volumes also shown in Fig. 7.

After the spatial constraints between the features of the component volumes have been defined, and the composite parameters have been defined in terms of the features of its component spatial primitives and parameters of its component composite volumes, it is possible to modify the spatial characteristics of a component volume and have the spatial characteristics of the composite volume be automatically modified through spatial constraint propagation. This in turn, may cause other composite volume parameters to be updated, and so on.

For composite volumes, the important dimensional parameters and their relationships to their component spatial primitives and composite volumes have to be defined at the time that the composite volume is constructed. At first, this requirement appears to be unrealistic for any reasonably complicated composite. However, in much the same way that one can define libraries of spatial primitives (blocks, cones, etc.), it is possible to define classes of commonly used composite volumes and predefine a lot of the parameters in terms of the features of their component volumes. If one uses one of these predefined classes of composite volumes, or takes the time to define a class of composite volumes, then form-function constraints can then be expressed between this composite volume and a functional subsystem. As will be discussed in a later section, form-function constraints can be used to specify how the shape of an object should change due to a change in its function, and vice-versa.

Hence, the spatial primitive and composite volume provide us with a mechanism for describing the shape, location and orientation of the physical objects and assemblies of physical objects that make up a building. Note, that there is no requirement that each spatial primitive be linked to functional primitive. Composite volumes may be linked to a functional primitive meaning that the individual spatial primitives do not provide function until they are aggregated into a composite.

4.3.3. Spatial Systems and Subsystems

The spatial system objects are intended to represent abstract spaces within a building such as floors, wings, rooms, etc, and their hierarchical and spatial interrelationships, e.g., wings are made up of rooms. Spatial system objects consist of a set of parameters that describe the abstract space (such as area, or length, width and height, and location), the function of the

space (making the spatial system a special kind of functional system description), and the spatial objects (primitives, composite volumes, or spatial subsystems) that are within the boundaries of this abstract space. It is important to note here that the spatial systems and subsystems are not represented using the underlying geometric modeller, as are the spatial primitives.

Similar to the way that the high-level functional systems are defined in the early part of design and refined into subsystems and functional primitives, the spatial system objects are defined early in the design process and continually refined. Eventually, each leaf of the spatial system decomposition (e.g., a room or hallway) is identified with a collection of spatial primitives and composite volumes. Also, similar to the way the attributes of a functional system are related to its components, the attributes of a spatial system are related to the attributes of its components through constraints. Although these constraints need not be represented, if they are not, the consistency of the levels of abstract spatial description can not be maintained. In other words, if the relationship between a parameter of an abstract space (e.g., width of a room) is not related to the physical objects that compose it (e.g., the locations of the inside surfaces of two of the four walls), then the abstract description of the room space may later become inconsistent with the wall locations if either wall is moved. However, even without constraints, the combination of the spatial system, composite volume and spatial primitive mechanisms facilitates the description of both abstract and detailed spatial description.

For an example of the three levels of spatial description, see Fig. 8.

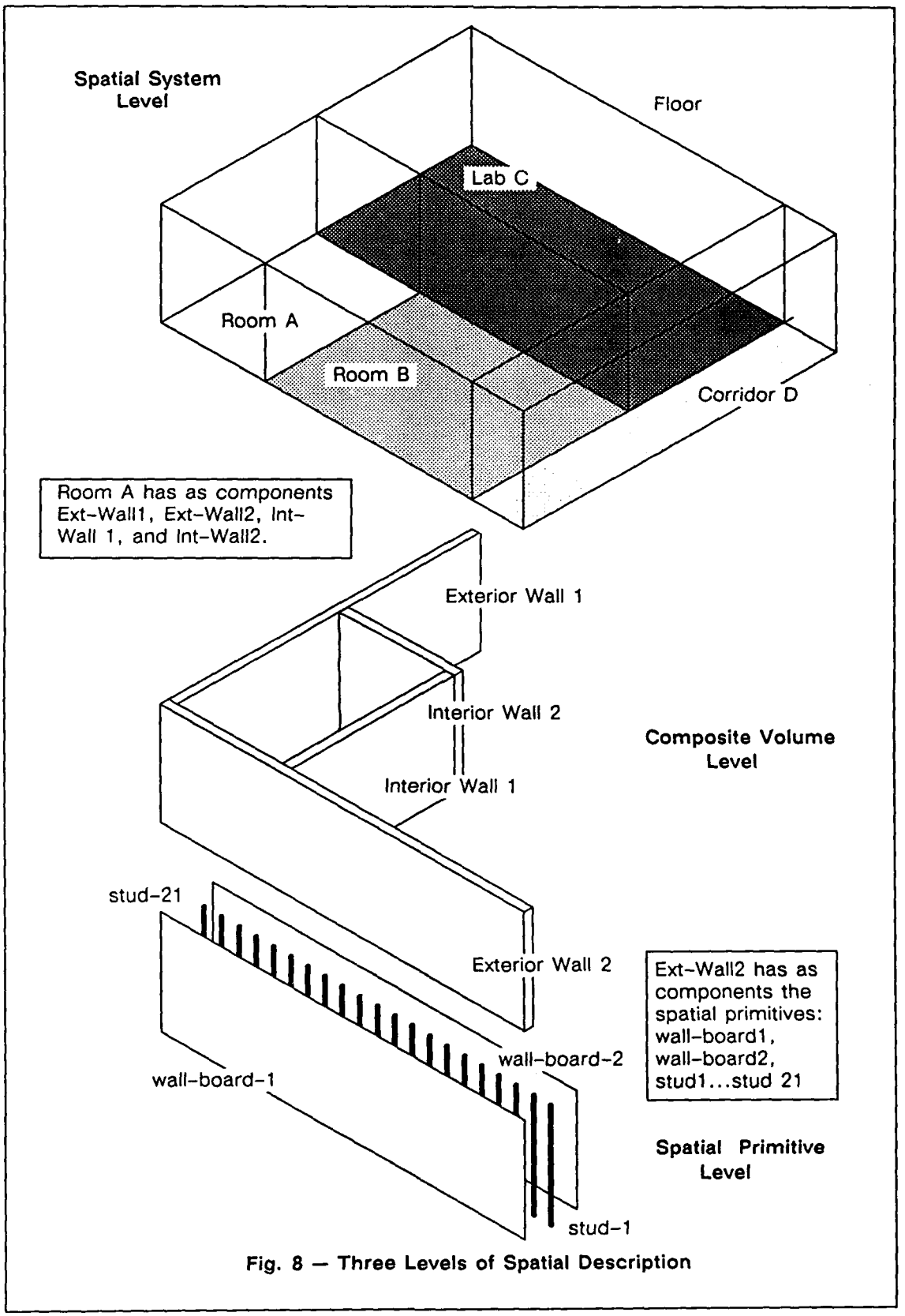


Fig. 8 — Three Levels of Spatial Description

4.4. Constraints

As illustrated in the previous sections describing functional and spatial decomposition, the "strings" that tie the levels of spatial and functional decomposition together are constraints. There are several types of constraints used within this model: functional constraints, spatial constraints, and form-function constraints. These three types of constraints provide a mechanism for describing the interrelationships of the objects within this building model.

4.4.1. Functional Constraints

Functional constraints relate the attributes of functional systems, subsystems and primitives. They are unidirectional in that they predefine the dependent and independent attributes. If any of the independent attributes are changed, the dependent attribute is automatically updated. For example, consider the set of functional primitives and the functional subsystem shown in Fig. 9. The definition of a functional constraint consists of an algebraic expression (in terms of

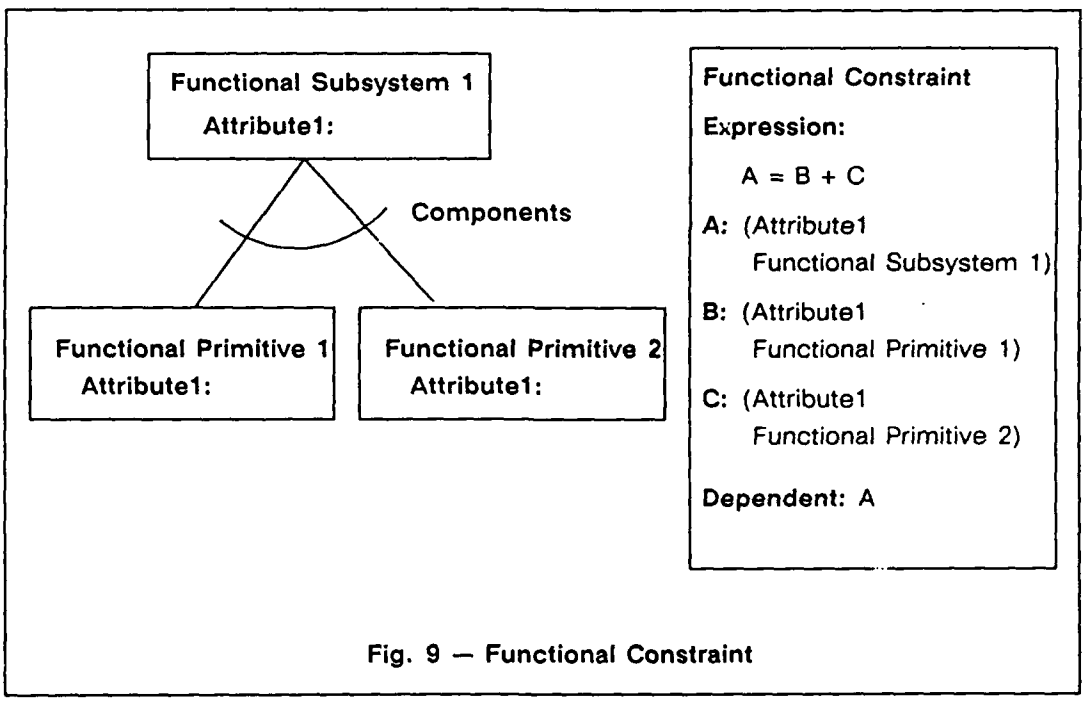


Fig. 9 — Functional Constraint

local variables), a definition of the mappings of these local variables to the attributes of the functional system and primitives, and an identification of the dependent variable. By default, all attributes in the expression not indicated as the dependent variable are treated as independent variables within this constraint. However, independent variables in this constraint may in fact be dependent variables in another constraint. When a functional constraint is defined, active values (also known as demons and discussed in Section 2.2.) are set up to watch the attributes associated with the independent variables. These active values act as

representatives of the constraints and notify the constraint if any of the attribute values associated with independent variables of the constraint are modified. Upon notification of a change in its independent variables, the constraint "fires" and modifies the value of the attribute associated with the dependent variable using the provided constraint expression. Once this attribute is changed, it may cause another constraint to be activated, and in this way the changes to functional attributes are propagated through the functional decomposition.

Note that there is an implicit assumption that the functional constraints being defined are equality constraints and are described in a non-circular fashion. If a circular set of constraints are defined, an infinite loop would occur. At this time, no mechanism exists to prevent the circular definition of constraints. However, it is possible to add active values to the constraint object itself that, during constraint definition, could parse a specified constraint expression and trace through all existing constraints to ensure that a path does not exist between the defined dependent variable and independent variables. The exclusive use of equality constraints in defining functional constraints is not deemed a limitation, because it is envisioned that these constraints represent the behavioral relationships of the components. The use of inequalities most often occurs when defining behavioral limitations, such as those provided by design standards. In dealing with design standards, a different class of constraints from these functional constraints will have to be defined and handled separately. Current research is being performed to define and manipulate standards-derived constraints on the attributes, but does not directly impact on the representational objectives of this model.

Requiring that the constraints be non-circular presents a particular problem in that we require that unidirectional constraints be defined from the components to the aggregate and from the aggregate to the functional components. Remember that the latter constraints are needed in order to ensure consistency between the functional subsystem and its components when the functional subsystem attribute is modified due to a relationship to another functional subsystem. This problem can be solved by giving the active values that notify constraints of a change in a dependent variable a bit of "intelligence." The constraints defined between a functional subsystem and its components and between the components and the functional system can be thought of a "inverses" of each other. This is information that the active values can use to temper their reactions. Another piece of information that is needed by the active values is which constraint led to the change of the attribute value that they are watching. Armed with these two pieces of knowledge, the active value simply checks to see if the value is changed by a constraint that is an inverse of the constraint to which the active value is reporting. If this is the case, the active value simply remains silent. If the change occurs as a result of a constraint not known to be an inverse, the active value notifies its constraint.

4.4.2. Spatial Constraints

There are three types of spatial constraints: constraints between the features of the spatial primitives, constraints between the features and the parameters of the spatial object, and constraints between the parameters of the spatial objects.

Spatial feature constraints — relate the features of one spatial object to the features of another. Because of the nature of spatial objects, the nature of a spatial constraint is to describe how one feature should be moved when another is moved. For example, consider the two cubes shown in Fig. 10., where Cube1 has a feature defined as BottomSurface and

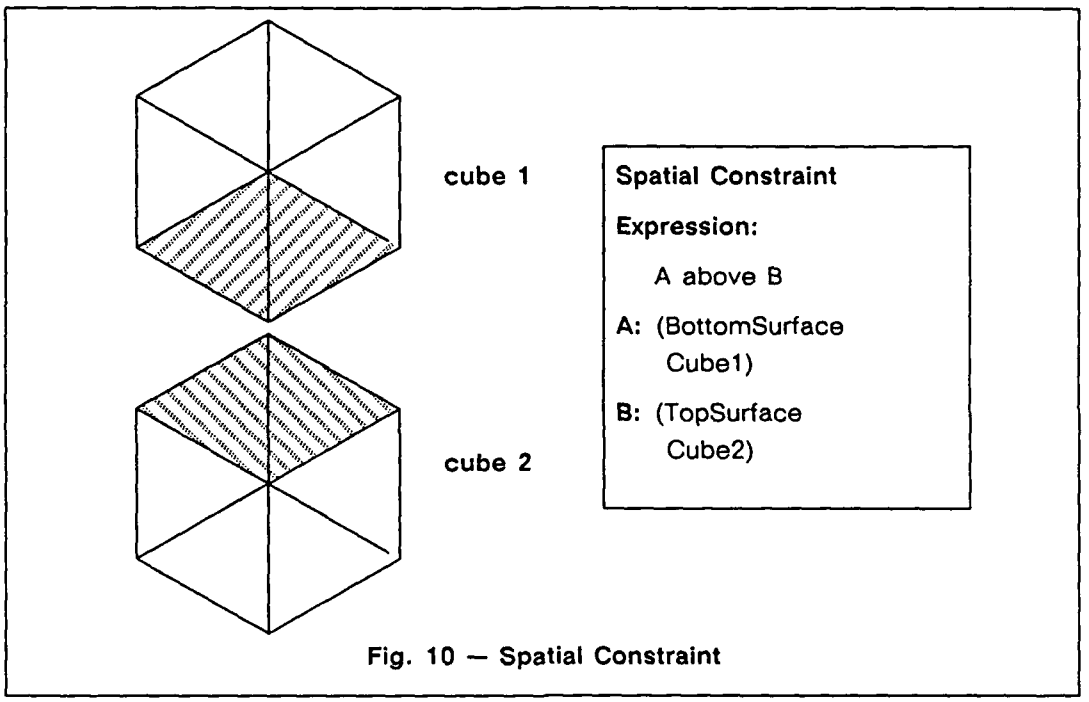


Fig. 10 — Spatial Constraint

cube2 has a feature defined as TopSurface. If the bottom surface of Cube1 is to remain above the top surface of Cube2, a spatial constraint can be written between the two features of these two cubes that maintains this positioning as shown in Fig. 10. If Cube2 is moved such that its top surface is above the bottom surface of Cube1, the bottom surface of Cube1 will be moved exactly above the top surface of Cube2, as shown in Fig. 11. Note that the shape of Cube1 has been modified. If the vertical size of Cube1 was not to be changed, another constraint must be written between the bottom surface and the top surface of Cube1; the constraint would state that the top surface is to remain a fixed distance from the bottom surface.

All spatial constraints are of the form: <feature> <operator> <feature>. Features are parts of a spatial object as defined previously. Any feature that is moved becomes the "independent"

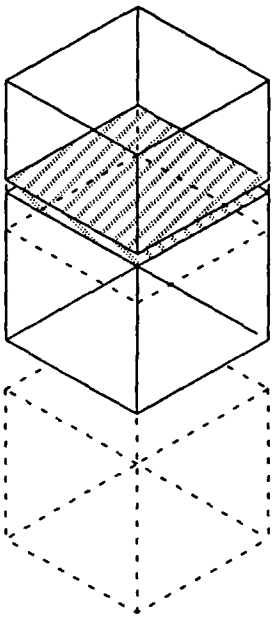


Fig. 11 — After Enforcement of Spatial Constraint

variable of any attached constraints and the other feature becomes the "dependent" variable. The operators that can be used are as follows:

above	where all vertices of the dependent feature are to have Y coordinates greater than or equal to those of the independent feature (this operator is the inverse of below);
below	where all vertices of the dependent feature are to have Y coordinates less than or equal to those of the independent feature (this operator is the inverse of above);
east-of	where all vertices of the dependent feature are to have X coordinates greater than or equal to those of the independent feature (this operator is the inverse of west-of);
west-of	where all vertices of the dependent feature are to have X coordinates less than or equal to those of the independent feature (this operator is the inverse of east-of);
north-of	where all vertices of the dependent feature are to have Z coordinates greater than or equal to those of the independent feature (this operator is the inverse of south-of);
south-of	where all vertices of the dependent feature are to have Z coordinates less than or equal to those of the independent feature (this operator is the inverse of north-of); and

attached-to where all vertices of the dependent feature are to be moved the same as that of the independent feature (this operator is the inverse of itself).

The last operator, attached-to, can be used for representing both attachment constraints and fixed-distance constraints. If two features are separated by a distance, but constraint to be attached, they will maintain their original separation when either is moved.

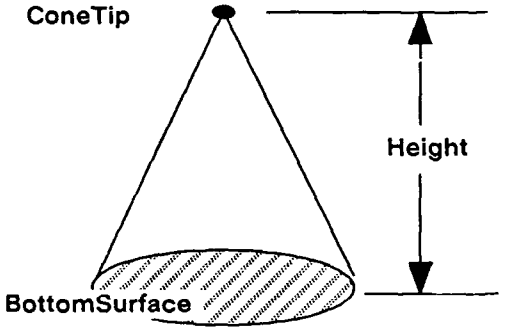
Because of the structure of the spatial constraint, the spatial constraints are not unidirectional, as is the case for functional constraints. If the bottom surface of Cube1 was moved downward, instead of the top surface of Cube2 being move up as in the previous example, the same constraint would be activated. The only difference is that the bottom surface feature of Cube1 would be the independent variable and the top surface feature of Cube2 would be the dependent variable. Cube2 would be shortened.

Whenever a spatial constraint is defined between two features, active values are attached to both of the features to watch for any movement. If movement of one of the two features occurs, the constraint is notified, the dependent and independent features are identified, and the constraint expression is used to update the location of the dependent feature based on the movement of the independent feature. Like the functional constraints, the spatial constraints written between features must be such that there are no circularities that would cause an infinite loop. Again, this is accomplished, at least locally, by having the active value determine if feature movement is due to the propagation of the constraint to which it reports. For example, if we define the constraint "A attached-to B", active values are set up to watch features A and B. If A is moved, then the constraint is notified by the active value watching A, which causes the constraint to move B in the same way A was moved. At that point, the active value watching B notices a change in the location of B, but it does not notify the constraint because B was moved as a result of the constraint to which the active value watching B reports.

The other way in which circularities can occur is if several constraints are written that start and end at the same feature, such as "A attached-to B", "B attached-to C", and "C attached-to A." This situation cannot be tolerated, and must be prevented. Like with the functional constraints, this type of circularity is not prevented, but could be by tracing the path of a new constraint with those already defined to ensure that no circular path exists. Thus, only non-circular set of constraints would be permitted.

Spatial feature/parameter constraints — are used to maintain consistency when parametric dimensions are used to generate the underlying solids model; they serve to relate the parameters to the resulting features. For example, given the cone shown in Fig. 6., a parameter/feature constraint of the form shown in Fig. 12. can be defined to watch these two

features for movement and recalculate the value of height accordingly.



Feature/Parameter Constraint

Expression:

$C = SQRT[(Bx - Ax)^2 + (By - Ay)^2 + (Bz - Az)^2]$

C: (THE Height of Cone1)

Ax: (THE x-location OF (THE BottomSurface OF Cone1))

.
.

Bz: (THE z-location OF (THE Tip OF Cone1))

Fig. 12 — Feature/Parameter Constraint

Note, that the feature/parameter constraints were hard-coded as active-values in their first implementation and were never actually expressed as formal constraints. However, there was nothing about the procedures that were written that would prevent them from being expressed in the above form.

Spatial parameter/parameter constraints — are simply algebraic constraints similar in form to the functional constraints described previously. This type of constraint exists between the parameters of the spatial systems and subsystems and maintain consistency between the levels of abstraction for spatial description.

4.4.3. Form-Function Constraints

A third type of constraint that can exist between the objects in this modelling environment are those between the functional and spatial decompositions — the form-function constraints. These constraints describe relationships between the function of a functional object and its spatial description represented by a spatial object (primitive, composite, or system). For example, the relationship between the diameter of a pipe (a spatial quantity) and its flow capacity (a functional quantity) could be represented as a form-function constraint. These constraints are quite similar to functional constraints, where they are algebraic constraints between the attributes of a functional system, subsystem or primitive and the dimensional parameters of a spatial object. Note, that these form-function constraints do not directly

interact with the spatial features. However, they indirectly impact on spatial features in that the dimensional parameters are dependent on feature location and visa-versa.

4.5. Integrated Model

So far, the functional and spatial decompositions have been described in detail, but separately. However, the object-oriented building model is actually an integration of these two decompositions as shown in Fig. 13. Each functional primitive is related to either a spatial primitive or a composite volume. For example, a beam object in the functional decomposition may be linked to a corresponding spatial primitive that represents a wide-flange shape having a certain length and made out of a-36 steel. However, it may alternatively be linked to a composite volume comprised of a wide-flange section and two plates. The links between the functional primitives and the spatial objects (shown in Fig. 13.) are shown as "is-a" links but do not connote class/subclass relationships as they normally do; the "is-a" link between the spatial object and the functional object permits the sharing of information (although only in one direction — from spatial to functional) between these two building descriptions via the inheritance capabilities of the object-oriented environment. For example, a functional system object inherits all of the dimensional attributes, features, and underlying solids model from its corresponding spatial object. At that point, all information about that object — function, form, location, etc., can be accessed, even though it is represented in a distributed form (function is separated from form in the model).

However, the unidirectional nature of the "is-a" relationship is actually inadequate for describing and implementing the linkage between functional and spatial descriptions. The linkage between these two descriptions should facilitate information retrieval in both directions — while within the spatial description, it should be possible to access functional attributes and attribute values and while within the functional description, it should be possible to access spatial attributes and attribute values. This is not really an "is-a" relation, but rather is an "is alternatively described by" relationship. Although no such relationship exists in current object-oriented environments, such a relationship can be simulated by defining two is-a links, one going in both directions. Although most current object-oriented systems will complain about a cyclic definition, it is possible in some to define such a cycle and inherit information in both directions. This type of relationship is vital to such a distributed representation of building information.

Hence, the two decompositions, spatial and functional, exist as separate decompositions, but are interrelated in several ways: by "is alternatively described by" relationships between functional objects and spatial objects, and form-function constraints in which the interdependencies of functional attributes and spatial attributes are defined. The benefits of

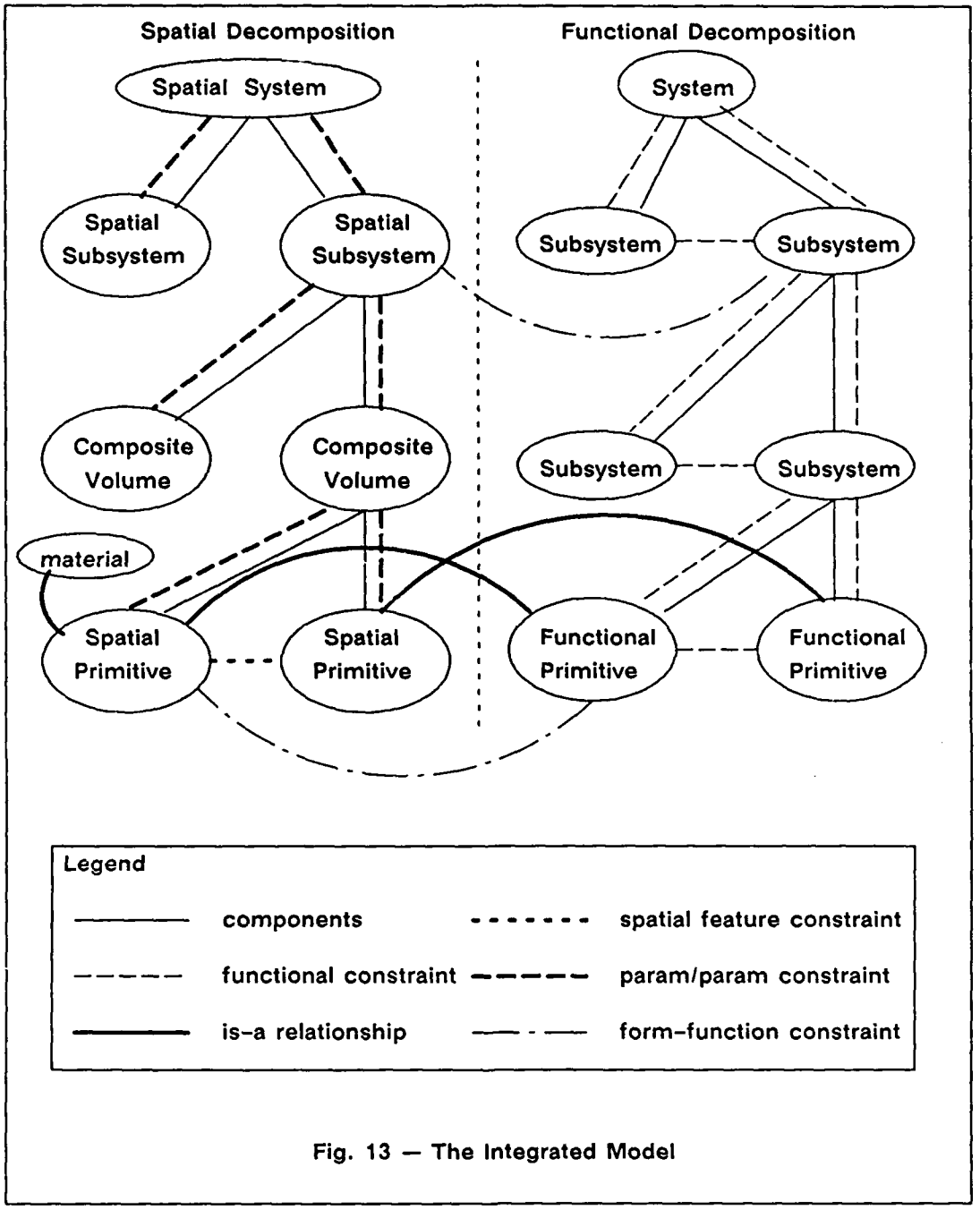


Fig. 13 — The Integrated Model

having the two decompositions separated is that one can describe the functionality of a building before committing to a specific form and vice-versa. Object-oriented systems that have both form and function described in the same object do not provide this flexibility.

5. SUMMARY AND RECOMMENDATIONS

What has been described in the proceeding pages is an object-oriented architecture of a building modelling environment that would be capable of representing, in a highly integrated manner, the functional and spatial decomposition of a constructed facility. Although small portions of this architecture have been implemented to test the concepts described, such as the description of spatial constraints between the features of a spatial objects and the the description of functional constraints between the attributes of functional systems, implementation of this modelling environment is nowhere near completion.

The architecture consists of collections of functional and spatial objects describing the various levels of functional and spatial decomposition of a building, which are interrelated via several types of constraints — functional, spatial and form-function constraints. The functional description is achieved using two levels: functional systems and functional primitives. The spatial description of the building is achieved using three levels: spatial primitives, composite volumes, and abstract spatial systems. The geometric description of the spatial primitive is also achieved using three levels: a complete underlying solids model, an identification of the important features of this underlying solids model, and a set of dimensional parameters that are related to the spatial positioning of the identified features.

Having such a modelling environment will facilitate integration of many of the tasks of the building design, construction and maintenance process. During design, the ability to express spatial and functional constraints between the various spatial and functional components of a building, respectively, will provide a means of communication between designers working concurrently on different parts of the problem. During construction, having both the functional and spatial decompositions of a building will allow construction managers to view the building from either perspective while decomposing the building into a set of construction activities. If construction tasks are to be automated, this model will act as the "mental image" of the structure and the interrelations of its parts that could be used by robotic construction activities. Finally, during renovation the model can be used to perform "what-if" types of analyses and to determine the effect on spatial and functional characteristics of the building to renovations and modifications. Without the explicit representation of spatial and functional interactions, achieved via constraints, all of the above mentioned activities are severely hindered. **This is one of the major problems with current CAD capabilities; they represent what the building looks like, but do not capture the functional and spatial interrelationships of the objects comprising the building.**

The proposed object-oriented model architecture described in this report has yet to be implemented, although most concepts have been tested for feasibility. From these concepts

tests, it was determined that complete implementation of this model will require a computing environment that tightly integrates the following capabilities (none currently exists and is why this architecture has not been fully implemented):

1) *Object-Oriented Programming.* For our testbed, we used KEE[7]. While KEE provided the requisite capabilities of object-oriented programming, it was too cumbersome and at times inflexible. What we need is an environment, for which we have the source code, that provides object-oriented programming capabilities and would allow us to integrate the other capabilities described below.

2) *Object-Oriented Solids Modelling.* For our testbed, we used VANTAGE[11], an object-oriented solids modeller developed at Carnegie Mellon University by Professor Takeo Kanade. While we do have the source to VANTAGE, it is implemented in an object-oriented environment other than KEE. Translation into KEE, although possible, was not performed. Instead, we attempted to have objects in KEE refer to the frames in the frame-based environment of VANTAGE. While this linkage worked for our testbed, it was extremely clumsy. From our experiences, we have concluded that the solids modeller must be implemented in the object-oriented environment, allowing the VANTAGE objects to actually be the spatial features to which our spatial primitives refer. This will solve one of our major problems: "How do the features of an object get automatically defined so that we can write constraints between them?"

3) *Constraint Expression and Management.* For our testbed, we started to implement a constraint expression and management facility, but found this to be a very large project all by itself. Because of the massive number of constraints envisioned to eventually exist in the model described in this report, a constraint management system implemented on top of an object-oriented environment is extremely vital to the successful implementation of this model. It should be noted here, that there is work being done within the Mechanical Engineering Department at the University of Illinois at Urbana-Champaign to develop a constraint expression and management system within a KEE-like environment for which source code is available.

6. ACKNOWLEDGEMENTS

This research was supported by the Army Research Office as part of the program of the University of Illinois Advanced Construction Technology Center.

7. REFERENCES

[1] Eastman, C. M., "The Representation of Design Problems and Maintenance of Their Structure", *Artificial Intelligence and Pattern Recognition in Computer-Aided Design*, Latombe, ed., North-Holland Publishing Company, 1978.

[2] Elam, S. L. and L. A. Lopez, "Knowledge Based Approach to Checking Designs for Conformance with Standards", Technical Report CESLRS No. 9, Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1988.

[3] Fenves S. J., U. Flemming, C. Hendrickson, M. L. Maher, and G. Schmidt, "An Integrated Software Environment for Building Design and Construction", in proceedings of The Fifth Conference on Computing in Civil Engineering, pp. 21-32, March 1988.

[4] Garrett, Jr., J. H. and S. J. Fenves, "A Knowledge-Based Standard Processor for Structural Component Design", *Engineering with Computers*, Volume 2, Number 4, pages 219-238, 1987.

[5] Howard, H. C., "KADBASE: An Expert System/Database Interface", in proceedings of The Fifth Conference on Computing in Civil Engineering, pp. 11-32, March 1988.

[6] Howard, H. C., R. E. Levitt, B. C. Paulson, J. G. Pohl and C. B. Tatum, "Computer Integration: Reducing Fragmentation in the AEC Industry", ASCE *Journal of Computing in Civil Engineering*, Volume 3, Number 1, pages 18-32, 1989.

[7] *KEE Software Development System User's Manual*, Document No. 3.0-U-1, IntelliCorp, Mountain View, CA, 1986.

[8] Keirouz, W. T. , D. R. Rehak and I. J. Oppenheim, "Development of an Object-Oriented Domain Model for Constructed Facilities", Engineering Design Research Center Report, Number EDRC-12-10-87, Carnegie Mellon University, Pittsburgh, PA, 1987.

[9] Law, K. H., M. K. Jouaneh and D. L. Spooner, "Abstraction Database Concept for Engineering Modelling", *Engineering with Computers*, No. 2, pp 79-94, 1987.

[10] Maher, M. L., and S. J. Fenves, "HI-RISE: An Expert System for the Preliminary Structural Design of High Rise Buildings", Technical Report R-85-146, Carnegie Mellon University, Department of Civil Engineering, November, 1984.

[11] *Vantage User's Manual*, Carnegie Mellon University, Pittsburgh, PA, 1987.

[12] Woodbury, R., "The Knowledge Based Representation and Manipulation of Geometry", Unpublished Ph.D. Dissertation, the Department of Architecture, Carnegie Mellon University, Pittsburgh, PA, 1987.

DISTRIBUTION LIST

This manuscript (ACTC Document #89-37-04) has been sent to the following persons and/or agencies.

Army Research Office, Library

U.S. Army Advisory Committee:

Dr. L.R. Shaffer, U.S. Army Construction Engineering Research Laboratory, Champaign, Illinois.
Dr. Fritz Oertel, U.S. Army Research Office, Research Triangle Park, North Carolina.
Dr. Lillian D. Wakeley, Waterways Experiment Station, Vicksburg, Mississippi.
Dr. Eugene Marvin, U.S. Army Cold Regions Research and Engineering Laboratory, Hanover, New Hampshire.
Dr. Robert Storer, Naval Civil Engineering Laboratory, Port Hueneme, California.
Dr. Giuliano D'Andrea, Bnt Laboratory, Watervliet, New York.
Dr. Margaret E. Roylance, Composite Materials Technology Laboratory, Watertown, Massachusetts.

Other

Jamie Florence, Manufacturing Technology Department of the Army, Warren, Michigan.
M. Kupperman, A. Epstein and Sons, International, Inc., Chicago, Illinois.
Bradley Posadas, Naval Civil Engineering Laboratory, Port Hueneme, California.
Phil Wager, Naval Civil Engineering Laboratory, Port Hueneme, California.
F. Kearney, U.S. Army Construction Engineering Research Laboratory, Champaign, Illinois.
Dr. Simon S. Kim, U.S. Army Construction Engineering Research Laboratory, Champaign, Illinois.
Professor Fred Moavenzadeh, Massachusetts Institute of Technology, Cambridge.
Mark E. Hollan, Naval Civil Engineering Laboratory, Port Hueneme, California.
Professor Lee Boyer, University of Illinois at Urbana-Champaign.
Professor John Melin, University of Illinois at Urbana-Champaign.
Professor Jim Lefter, University of Illinois at Urbana-Champaign.